REMARKS/ARGUMENTS

1. Claims 1-3, 5-8, 10-13, and 15 are Patentable Over the Cited Art

The Examiner rejected claims 1-3, 5-8, 10-13, and 15 as obvious (35 U.S.C. §103(a)) over Anonsen (U.S. Patent No. 7,082,433) in view of Chow (U.S. Patent No. 6,941,298). Applicants traverse

Independent claims 1, 6, and 11 recite concern translating a path expression in an object oriented query to a relational database outer join, said path expression comprising a navigation path through a relationship in a schema. The claims require: analyzing each path expression defined in each level of the object oriented query; identifying each path expression which can be a candidate for a translation to an outer join; ordering the path expression starting with path expression defined in a FROM clause, adding to the FROM clause path expression, each path expression identified as a candidate for a translation to an outer join, and making the ordered path expressions as input to a select operator for each level of the object oriented query; grouping the ordered path expressions sequentially based upon on a source-target dependency between ordered path expressions and based upon the identifications as a candidate for a translation to an outer join; creating a quantifier for each path expression, said quantifier comprising a variable representing a table in a relational database; replacing each grouped path expression with a corresponding quantifier and related table in a relational database; and completing a translation of the object oriented query to a relational query.

The Examiner did not cite to a specific section of the cited art as teaching the claim requirement of ordering the path expression starting with path expression defined in a FROM clause, adding to the FROM clause path expression, each path expression identified as a candidate for a translation to an outer join, and making the ordered path expressions as input to a select operator for each level of the object oriented query. The Examiner did cite to the Joinlist and col. 4, lines 14-19 as teaching making the ordered path expressions input to a selector operator. (Second Office Action, pgs. 4-5)

The cited joinlist 216 contains a list of explicit joins for an instance of the criteria. The join list identifies entities joined to other entities in the query. (Anonsen, col. 9, lines 14-19, col. 13, lines 44-50). The criteria allows users to define criteria that describe the entities being queried. (Anonsen, col. 7, lines 45-55) Although the cited joinlist has information on entities

involved in a join, the Examiner has not cited any part of Anonsen that teaches that the join list orders path expressions starting with a path expression defined in a FROM clause and adding to the FROM clause path expressions, where each path expression identified for a translation to an outer join and making the ordered path expressions as input to a select operator. There is no mention or teaching of adding to a FROM statement path expressions identified as a candidate for a translation to an outer join.

The cited col. 4 discusses an object relational data storage system in which stored data can be referred to in terms of entities or objects and their properties, rather than elements of the database schema. Nowhere does this cited col. 4 anywhere teach or suggest adding to the FROM clause path expressions identified as candidates for an outer join. Instead, the cited col. 4 discusses how data in a relational database may be referred to as objects and their properties.

The Examiner cited col. 5, line 65 to col. 6, line 18 and FIGs. 7-8 of Chow as teaching the claim requirements of ordering path expressions starting with a path expression defined in a FROM clause and adding to the FROM clause path expressions. (Second Office Action, pg. 6) Applicants traverse.

The cited cols. 5-6 of Chow discusses allowing a bean object developer to request multiple fields from multiple bean object types to be returned in a result set using a bean query. The bean developer can specify a select and from components for the query of the object beans. Although the cited Chow discusses a query for bean objects, nowhere is there any teaching or suggestion of the claim requirement of ordering the path expression starting with path expression defined in a FROM clause, adding to the FROM clause path expression, each path expression identified as a candidate for a translation to an outer join. There is no teaching or mention in the cited Chow of adding to the FROM clause expression, path expressions identified as candidates for translation toward an outer join. Instead, the cited Chow discusses a bean type request to allow a bean developer to request multiple fields from bean types to be returned in a result set using a beans query, by specifying a SELECT, FROM. and WHERE claims.

The Examiner cited col. 14, line 63 to col. 15, line 8 and col. 12, lines 16-25 of Anonsen with respect to the claim requirement of grouping the ordered path expressions sequentially based upon a source-target dependency between ordered path expressions and based upon the identifications as a candidate for a translation to an outer join. (Second Office Action, pg. 5) Applicants traverse.

The cited cols. 14-15 mentions that the query is parsed into a parse tree and a directed acyclic graph (DAG), such that the DAG contains the objects being joined and their joins to each other. The graph can be traversed to produce the correct joins in the correct order in relation to one other.

Although the cited cols. 14-15 mention using a DAG to produce joins in a "correct order", nowhere do the cited cols. 14-15 teach or mention grouping the ordered path expressions sequentially based upon on a source-target dependency between ordered path expressions and based upon the identifications as a candidate for a translation to an outer join. Instead, the cited cols. 14-15 mention generally using the DAG to produce the correct joins in the correct order and do not disclose the specific claim requirements concerning the ordering, such as sequentially based upon a source target dependency.

The cited col. 12 mentions that an expression forms a parse tree, where each box in the tree of FIG. 4c represents an object in memory and the lines represent references between the objects. Each expression object is assigned a component of the criteria object that can be accessed during translation. Each query would have completely parsed expressions.

Although the cited col. 12 discusses how expressions may form a parse tree and that each query has parsed expressions, nowhere is there any teaching or mention in the cited col. 12 of grouping ordered path expressions sequentially based upon a source-target dependency between ordered path expressions and upon the identification as a candidate for an outer join. Instead, the cited col. 12 mentions how expressions may form a parse tree.

Thus, the Examiner has not cited any part of Anonsen that that teaches or mentions grouping ordered path expressions sequentially based upon a source-target dependency between ordered path expressions and upon the identification as a candidate for an outer join.

The Examiner cited col. 15, lines 22-32 and col. 14, lines 48-62 of Anonsen as teaching the claim requirements of creating a quantifier for each path expression, said quantifier comprising a variable representing a table in a relational database. (Second Office Action, pgs. 5-6) Applicants traverse.

The cited col. 15 mentions that each node in the DAG created for the parse tree has directed edges to other nodes, each of which refers to an object to which the original object joins. The nodes have a unique identity referred to as a qualifier. The qualifier for a node is the object path taken to reach the object represented by the node.

Although the cited col. 15 mentions a qualifier, the cited qualifier does not teach or suggest the claimed quantifier that comprises a variable representing a table in a relational database. Instead, the cited "qualifier" is an object path taken to reach the object represented by the node, not a variable representing a table in a relational database as claimed.

The cited col. 14 mentions joining two objects by any arbitrary property on those objects by specifying which properties on either object to join and any arithmetic operators, relational operators, Boolean operators, unary operators, etc., as necessary. In this way, the developers may express the queries in terms of qualified object references combined with expressions separated by the operators. The qualified object references thus require implicit joins, since the joins are not explicitly stated.

Although the cited col. 14 discusses joining two objects and qualified object references that require implicit join references, nowhere is there any teaching or mention of a quantifier that comprises a variable representing a table in a relational database.

The Examiner cited col. 15, lines 11-21 as teaching the claim requirement of replacing each grouped path expression with a corresponding quantifier, representing a table in a relational database, and related table in a relational database. (Second Office Action, pg. 6)

The cited 15 mentions that once the parse tree is generated, translator component 13 in data access system 12 traverses the parse tree in post-fix order to build a directed acyclic graph (DAG) for the parse tree. Each node of the DAG represents an object within the join expression that is mapped to a different row in the relational database. As mentioned above, there are explicit joins which are specified by the developer. However, there are also implicit joins which are introduced because a property reference crosses the boundary between two objects that are mapped to different rows by class-table mapping 18 (shown in FIG. 1).

Although the cited col. 15 mentions a DAG representing an object within a join expression mapped to a different row in the relational database, there is no teaching or mention of the claim requirement of replacing each grouped path expression with a corresponding quantifier representing a table in a relational database.

Accordingly, applicants submit that claims 1, 6, and 11 are patentable over the cited art because the cited Anonsen and Chow do not teach or suggest all the claim requirements.

Claims 2, 3, 5, 7, 8, 10, 12, 13, and 15 are patentable over the cited art because they depend from one of claims 1, 6, and 11. The following claims provide additional grounds of patentability over the cited art.

2. Claims 4, 9, and 14 are Patentable Over the Cited Art

The Examiner rejected claims 4, 9, and 14 as obvious (35 U.S.C. §103(a)) over Anonsen, Chow, and further in view of Touma (U.S. Patent No. 6,160,549). Applicants traverse.

These claims are patentable over the cited art because they depend from one of claims 2, 7, and 12, which are patentable over the cited art for the reasons discussed above. Moreover, the additional requirements of these claims provide further grounds of patentability over the cited art for the following reasons.

Claims 4, 9, and 14 depend from claims 2, 7, and 12 and additionally require that the optimization identifies a quantifier as a candidate for a translation to an inner join if a LIKE, IN, or BETWEEN operator exists in a WHERE clause containing a corresponding path expression.

The Examiner cited col. 7, lines 23-54 of Touma as teaching the additional requirements of this claim. (Second Office Action, pg. 9) Applicants traverse.

The cited col. 7 mentions a link data object to establish parent-child relationships between two or more queries or groups that are made using matching columns. A link enables a user to specify an SQL clause and match condition in a relationship between two columns beyond the standard SQL equijoin. A link, i.e., parent-child relationship, causes the child query to be executed once for each instance of its parent group. The link condition includes equal to, less than, less than or equal to, not equal to, greater than, greater than or equal to, like, and not like. The "like" operator results in a true when the value in one column matches the pattern (including wildcard positions) in the other column. The "not like" operator results in a true condition when the value in one column does not match the pattern (including wildcards) in the other column.

Although the cited col. 4 discusses a link to specify a relationship between objects, including a like operator, nowhere is there any teaching or suggestion that the optimization identifies a quantifier as a candidate for a translation to an inner join if a LIKE, IN, or BETWEEN operator exists in a WHERE clause containing a corresponding path expression.

Accordingly, claims 4, 9, and 14 provide additional grounds of patentability over the cited art because the additional requirements of these claims are not taught or suggested in the cited art.

Conclusion

For all the above reasons, Applicant submits that the pending claims 1-15 are patentable over the art of record. Applicants have not added any claims. Nonetheless, should any additional fees be required, please charge Deposit Account No. 09-0460.

The attorney of record invites the Examiner to contact him at (310) 553-7977 if the Examiner believes such contact would advance the prosecution of the case.

Dated: August 21, 2007

By: /David Victor/
David W. Victor
Registration No. 39,867

Please direct all correspondences to:

David Victor
Konrad Raynes & Victor, LLP
315 South Beverly Drive, Ste. 210
Beverly Hills, CA 90212
Tel: 310-553-7977
Fax: 310-556-7984